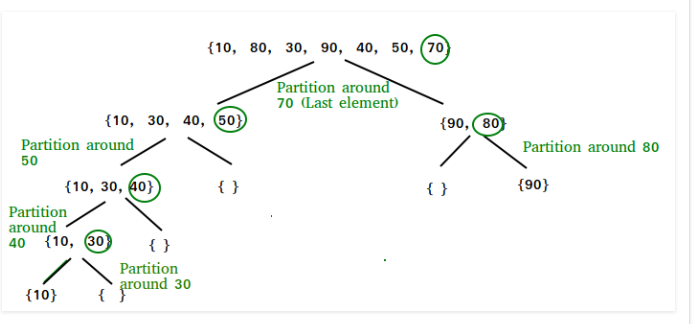


**Quick Sort:-**

Quicksort is a [divide and conquer algorithm](https://en.wikipedia.org/wiki/Divide_and_conquer_algorithm). Quicksort first divides a large array into two smaller sub-arrays: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays.

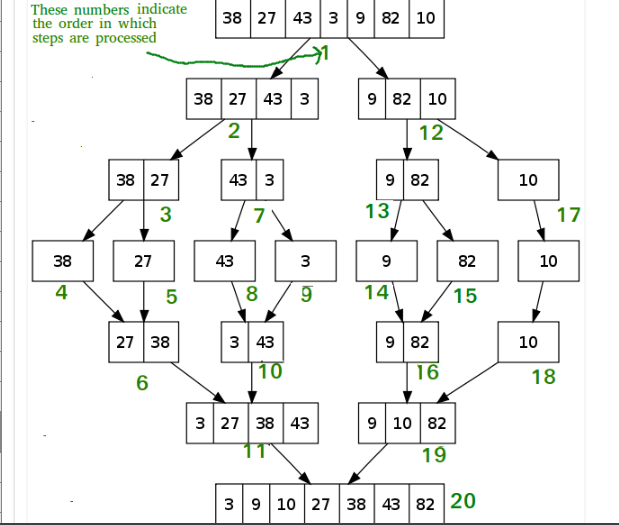
The steps are:

1. Pick an element, called a *pivot*, from the array.
2. *Partitioning*: reorder the array so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it (equal values can go either way). After this partitioning, the pivot is in its final position. This is called the *partition* operation.
3. [Recursively](https://en.wikipedia.org/wiki/Recursion_(computer_science)) apply the above steps to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values



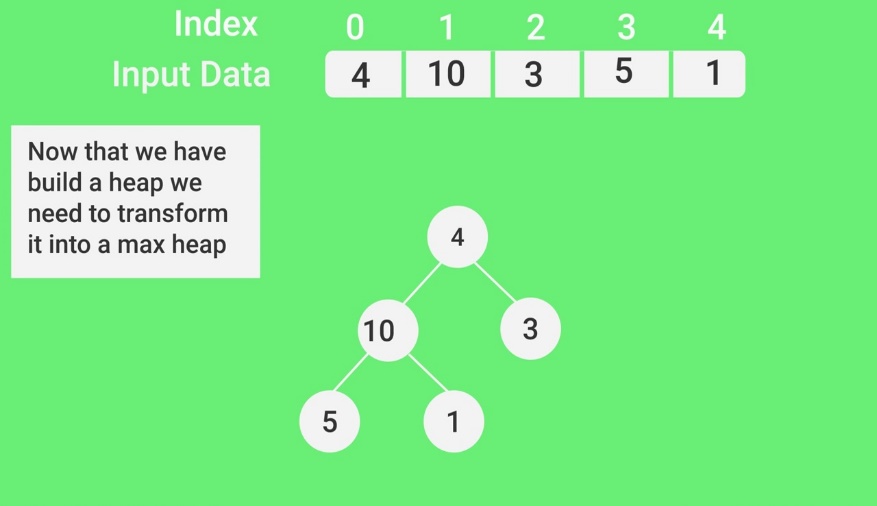
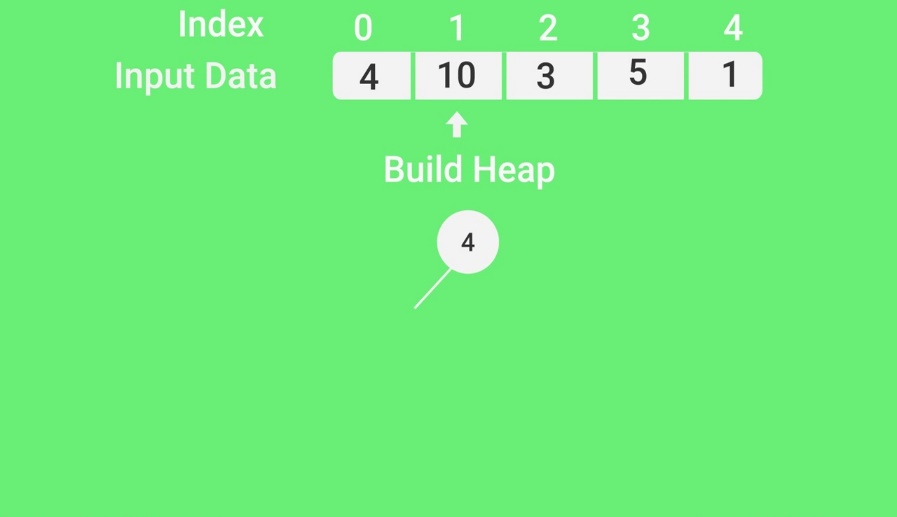
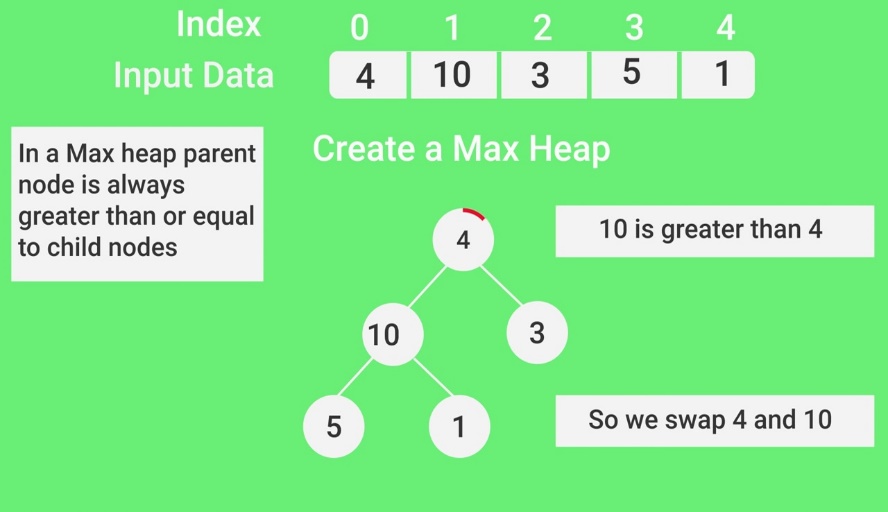
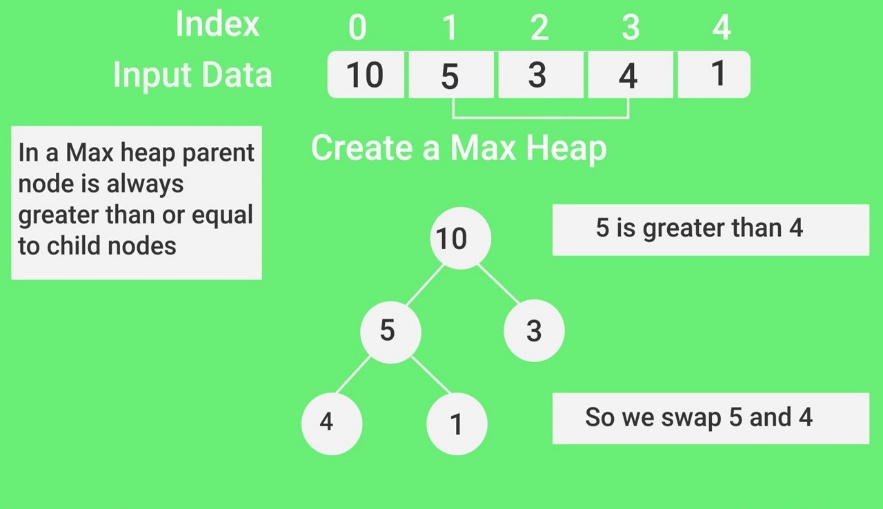
**Merge Sort:-**

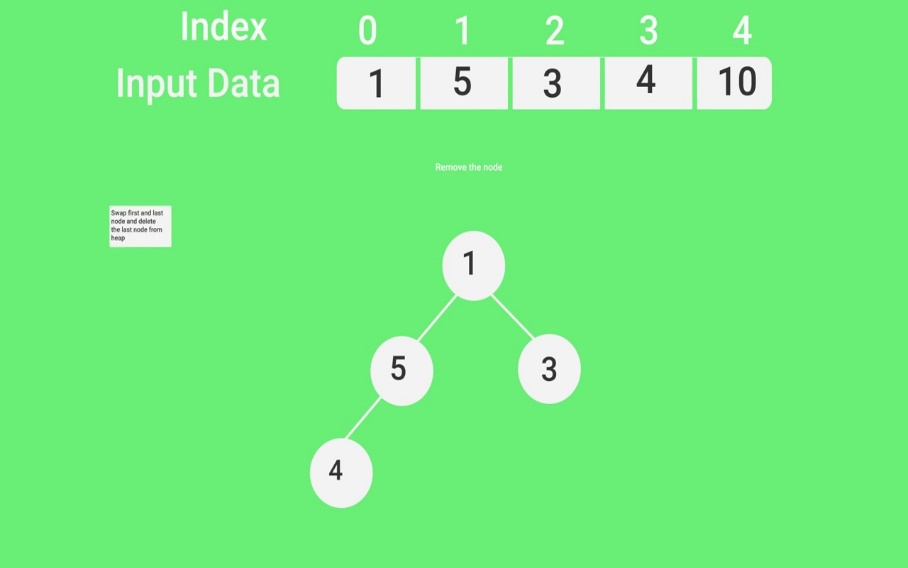
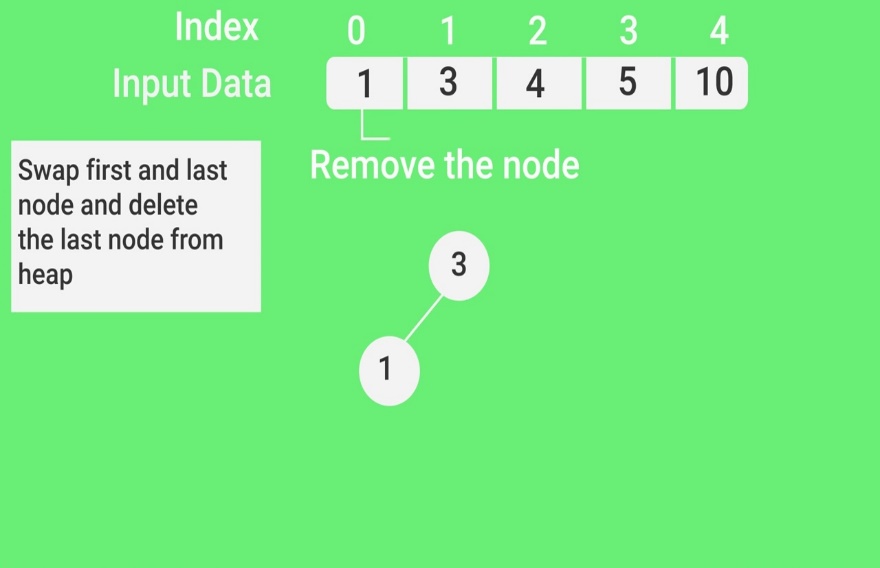
Like [QuickSort](https://www.geeksforgeeks.org/quick-sort/), Merge Sort is a [Divide and Conquer](https://www.geeksforgeeks.org/divide-and-conquer-introduction/) algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.



**Heap Sort:-**

Heap sort is a comparison based sorting technique based on Binary Heap data structure. It is similar to selection sort where we first find the maximum element and place the maximum element at the end. We repeat the same process for remaining element



**Smooth Sort:-**

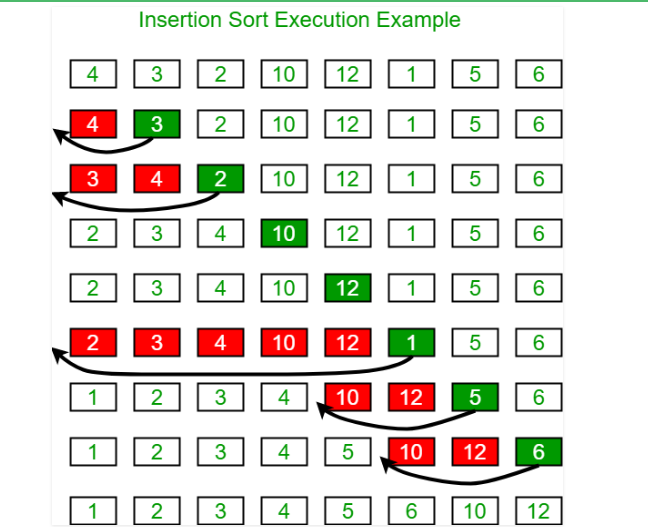
Like [heapsort](https://en.wikipedia.org/wiki/Heapsort" \o "Heapsort), smoothsort builds up an [implicit](https://en.wikipedia.org/wiki/Implicit_data_structure) heap data structure in the array to be sorted, then sorts the array by repeatedly extracting the maximum element from that heap. Unlike heapsort, which places the root of the heap and largest element at the beginning (left) of the array before swapping it to the end (right), smoothsort places the root of the heap at the right, already in its final location. This, however, considerably complicates the algorithm for removing the rightmost elem

**Bubble Sort:-**

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

**Example:**  
First Pass:  
( 5 1 4 2 8 ) –> ( 1 5 4 2 8 ), Here, algorithm compares the first two elements, and swaps since 5 > 1.  
( 1 5 4 2 8 ) –>  ( 1 4 5 2 8 ), Swap since 5 > 4  
( 1 4 5 2 8 ) –>  ( 1 4 2 5 8 ), Swap since 5 > 2  
( 1 4 2 5 8 ) –> ( 1 4 2 5 8 ), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

Second Pass:  
( 1 4 2 5 8 ) –> ( 1 4 2 5 8 )  
( 1 4 2 5 8 ) –> ( 1 2 4 5 8 ), Swap since 4 > 2  
( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )  
( 1 2 4 5 8 ) –>  ( 1 2 4 5 8 )  
Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

**Insertion Sort:-**

**Selection Sort:-**

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning

**Basic Array:-**

An array is a set of values, which are termed elements, that are logically related to each other. ... Similarly, an array may consist of a student's grades for a class; each element of the array is a single grade. It is possible individual variables to store each of our data items

**Dynamic Array:-**

In computer science, a dynamic array, growable array, resizable array, dynamic table, mutable array, or array list is a random access, variable-size list data structure that allows elements to be added or removed. It is supplied with standard libraries in many modern mainstream programming languages.

**Singly linked list:-**

Singly Linked Lists are a type of data structure. In a singly linked list, each node in the list stores the contents and a pointer or reference to the next node in the list. It does not store any pointer or reference to the previous node. ... The last node in asingle linked list points to nothing.

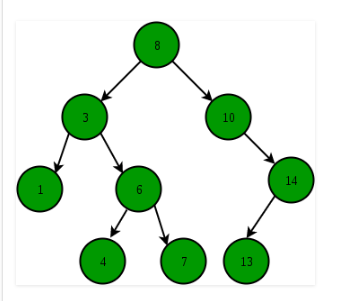
**Doubly Linked List:-**

Doubly linked list. In computer science, a doubly linked list is a linked data structure that consists of a set of sequentially linked records called nodes. Eachnode contains two fields, called links, that are references to the previous and to the next node in the sequence of nodes.

**Hash table:-**

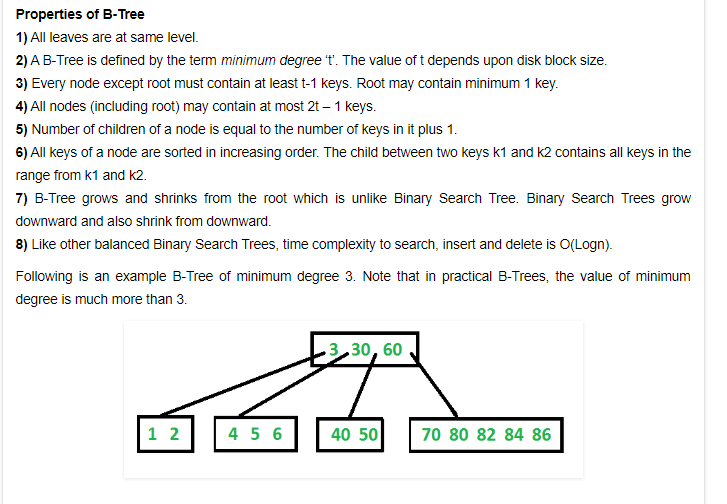
Hash Table is a data structure which stores data in an associative manner. In hash table, the data is stored in an array format where each data value has its own unique index value.

**Binary Search Tree:-**

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties − The left sub-tree of a node has a key less than or equal to its parent node's key. The right sub-tree of a node has a key greater than to its parent node's key.

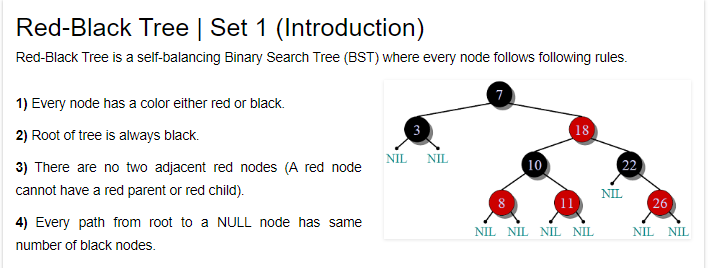
**B Tree:-**

B-Tree is a self-balancing search tree. In most of the other self-balancing search trees (like [AVL](https://www.geeksforgeeks.org/avl-tree-set-1-insertion/) and Red-Black Trees), it is assumed that everything is in main memory. To understand the use of B-Trees, we must think of the huge amount of data that cannot fit in main memory. When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to main memory access time. The main idea of using B-Trees is to reduce the number of disk accesses.

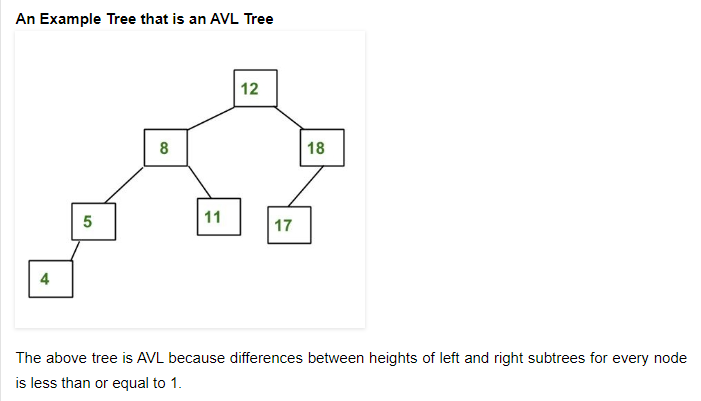


**Red-Black Tree:-**

A red–black tree is a kind of self-balancing binary search tree in computer science. Each node of the binary tree has an extra bit, and that bit is often interpreted as the color (red or black) of the node. These color bits are used to ensure the tree remains approximately balanced during insertions and deletions.



**AVL Tree:-**

 A balanced binary search tree where the height of the two subtrees (children) of a node differs by at most one. Look-up, insertion, and deletion are O(log n), where n is the number of nodes in the tree.

